Análisis comparativo de los lenguajes de programación de PLC definidos en la norma IEC 61131-3

Jhonatan Arias Polanco

Universidad Tecnológica de Pereira

Programa de ingeniería eléctrica

Pereira, Risaralda, Colombia

2019

Análisis comparativo de los lenguajes de programación de PLC definidos en la norma IEC 61131-3

Jhonatan Arias Polanco

Proyecto presentado como requisito para optar al título de:
Ingeniero Electricista

Director: MSc. Duberney Murillo Yarce
Programa de Ingeniería Eléctrica

Universidad Tecnológica de Pereira
Pereira, Risaralda, Colombia
2019

Agradecimientos

Hoy quiero expresar mis más sinceros agradecimientos. Primero a Dios quien siempre ha estado a mi lado, no me desamparo y fue mi guía, a toda mi familia especialmente a mi padre Luis Evelio Arias y mi madre Liliana Polanco por su apoyo incondicional y paciencia. Agradezco por su colaboración durante todos mis estudios; a los ingenieros Duberney Murillo Yarce, director de mi trabajo de grado, y Raúl Antonio Rodríguez Osorio por el acompañamiento incondicional que me brindó.

Agradezco también a todos mis docentes, compañeros y demás personas que de una manera u otra estuvieron conmigo a lo largo de este proceso.

Contenido

1. Automatización Industrial 6
2. Software 8
3. Objetivos 21
   3.1 Objetivo General 21
   3.2 Objetivos específicos 21
4. Planteamiento del problema 22
5. Solución del automatismo 23
6. Análisis de resultados 33
7. Conclusiones 34
8. Trabajos futuros 36
9. Bibliografía 37

Contenido de Figuras

Figura 1 Menú de automatismos 8
Figura 2 Configuración del POU 9
Figura 3 Ventana general Codesys V2.3 9
Figura 4 Crear nuevo POU 10
Figura 5 Librería Codesys V2.3 11
Figura 6 Contador CTU 11
Figura 7 Temporizador TON 12
Figura 8 Bloque de función BLINK 12
Figura 9 Crear visualización 13
Figura 10 Creación rectángulo y cambio de colores 14
Figura 11 Variable para el cambio de color 14
Figura 12 Proceso del problema de ejemplo 15
Figura 13 Visualización del ejemplo de simulación en Codesys V2.3 16
Figura 14 Lenguaje de programación SFC o GRAFCET 17
Figura 15 Lenguaje de programación LD o Ladder 18
Figura 16 Lenguaje de programación FBD 19
Figura 17 Lenguaje de programación ST 19
Figura 18 Lenguaje de programación IL 20
Figura 19 Fluidsim 21
Figura 20 Visualización de la simulación 24
Figura 21 Programación en GRAFCET 25
Figura 22 Acciones de la etapa 5 25
Figura 23 Programación en texto estructurado 27

Figura 24 Programación en lista de instrucciones 28
Figura 25 Programación en Ladder 29
Figura 26 Fragmento del código en diagrama de bloques 30
Figura 27 Programa principal PLC_PRG 31
Figura 28 PLC Festo CPX-CEC-C1 31
Figura 29 Programación en el programa PLC_PRG 32
Figura 30 Visualización circuito neumático y eléctrico en Fluidsim 33

1. Automatización Industrial

La automatización industrial surge dado que las empresas están en constante búsqueda de obtener altos volúmenes de producción, seguridad, calidad y eficiencia en el desarrollo de los procesos industriales. Por otra parte, es una gran herramienta para los operadores ya que se puede prescindir de la operación humana en trabajos que representan un riesgo para la integridad física de los trabajadores, de la misma manera las empresas pueden disminuir costos operativos.

En sus inicios, la automatización se implementaba mediante lógica cableada haciendo uso de elementos como son los contactos auxiliares de relés electromecánicos, contactores de potencia, relés temporizadores, relés contadores, entre otros. Después, surgieron los sistemas embebidos y se desarrolló la lógica programada.

Existen diferentes tipos de controladores lógicos lo cuales se encuentran divididos en dos grandes grupos que son los controladores con unidad operativa y controladores sin unidad operativa, teniendo en cuenta que ambos tipos de controladores tienen sus respectivas sub-divisiones. Los controladores lógicos sin unidad operativa a su vez están conformados por los combinacionales y secuenciales. El conjunto de combinacionales lo constituyen los controladores por cableado y los programables. Por otro lado, los secuenciales pueden ser asíncronos o síncronos; los controladores secuenciales síncronos de la misma manera que el conjunto de combinacionales está establecido por los controladores con cableado y programables, aunque los controladores programables síncronos se dividen por su arquitectura que es fija o configurable. Así mismo los controladores lógicos con unidad operativa cuentan con controladores con unidad lógica y controladores basados en procesadores. Por último, los controladores basados en procesador esta creado por ordenadores industriales, microprocesadores y Controladores Lógicos Programables (PLC por sus siglas en inglés).

El PLC es un dispositivo electrónico que permite programar y controlar procesos en tiempo real y son los más usados en la industria en general. Debido a la gran demanda de PLC a nivel mundial, hoy en día existen muchos fabricantes de PLC y cada fabricante ha definido su propio lenguaje

de programación de PLC. Algunos de los fabricantes más conocidos de PLC son Festo, Mitsubishi, Siemens, Schneider Electric, entre otros.

Dado que hoy en día hay una gran variedad de fabricantes y cada uno ha definido su propio lenguaje de programación de PLC, se creó un estándar internacional con el ánimo de normalizar todo lo relacionado con esta tecnología, la norma IEC61131 y, en particular para los lenguajes de programación se desarrolló el documento independiente IEC 61131-3. Los lenguajes del estándar IEC 61131-3 son el diagrama de funciones secuenciales (SFC) también conocido como GRAFCET, diagrama de bloques de funciones (FBD), Diagrama de tipo escalera o diagrama de contactos (LD o Ladder), texto estructurado (ST), y lista de instrucciones (IL).

Los lenguajes del estándar IEC 61131-3 se pueden clasificar en dos ramas, los de tipo gráfico y los de tipo textual. Los lenguajes gráficos están conformados por los lenguajes LD, SFC y FBD, por otro lado los lenguajes de tipo textual están constituidos por los lenguajes de ST e IL.

El Diagrama de Funciones Secuenciales es un lenguaje de programación gráfico el cual consta de etapas y transiciones, una etapa puede estar en estado activa o inactiva, cuando la etapa se encuentra activa, las instrucciones asociadas son ejecutadas hasta que dicha etapa se vuelva inactiva.

De igual manera el Diagrama de Bloques de Funciones es un lenguaje gráfico, y es conveniente su uso cuando no hay ciclos; Consta de una aritmética gráficamente conectada, booleana y otros tipos de elementos funcionales, una de sus mayores ventajas es que se puede visualizar con mayor facilidad las operaciones en secuencias.

El diagrama de Contactos también es un lenguaje gráfico y es el más común para la programación de los PLC, este lenguaje se caracteriza por su relación con el diseño de un circuito de lógica cableada, permitiendo incluso, desarrollar la misma lógica en la elaboración de un programa para los controladores lógicos.

Por otro lado el Texto Estructurado y la lista de instrucciones son lenguajes de programación basados en texto, el lenguaje ST es muy similar a lenguajes como Pascal y el C y utiliza tres tipos de estructuras las cuales son secuencias, instrucción condicional (selección) y repetición condicional (interacción); Por último, cabe mencionar que el lenguaje IL es similar al lenguaje ensamblador o mnemónico.

2. Software

Actualmente existen muchos paquetes de software de programación para PLC creados por cada fabricante, generalmente para ser usados solamente en sus dispositivos, otros han sido desarrollados para vincular múltiples fabricantes; No todos los paquetes de software permiten programar en todos los lenguajes del estándar IEC-61131-3.

Codesys es un software de programación de PLC que permite el uso de los lenguajes del estándar IEC 61131-3. Adicionalmente permite programar con una lista muy extensa de fabricantes de PLC´s como son Bosch, Schneider Electric, Festo, IFM entre otros.

Para crear un nuevo proyecto en Codesys V2.3 (en el idioma inglés) se hace clic en *New* e inmediatamente aparece un cuadro de dialogo como el que se muestra en la figura 1, posteriormente se procede a seleccionar el autómata al que será conectado y se acepta el procedimiento. La opción *3S CoDeSys PLCWinNT V2.4* es un PLC virtual creado para realizar simulaciones.

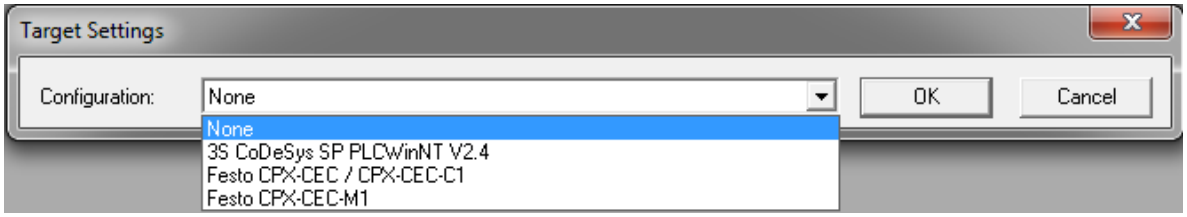

Figura 1 Menú de automatismos

Se selecciona un nuevo POU (Program Organization Unit), en esta ventana se escoge el tipo de lenguaje de programación, el tipo de POU (programa, bloque de función, función) y, se asigna el nombre del proyecto, teniendo en cuenta que el programa principal se debe nombrar PLC_PRG. En la figura 2 se observan los lenguajes del estándar IEC 61131-3 y también CFC que no está normalizado, es decir que no se encuentra regido bajo la norma IEC 61131-3, pero el software Codesys lo ofrece como alternativa al FBD con una estructura de programación no segmentada.

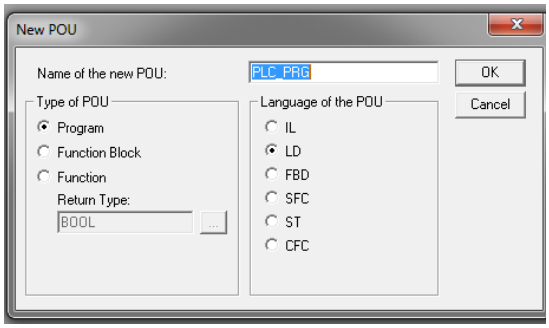

Figura 2 Configuración del POU

Luego de ser creado el programa aparecerá la ventana principal en la que se cuenta con un menú de trabajo. En la figura 3 se muestra el menú de trabajo de Codesys V2.3

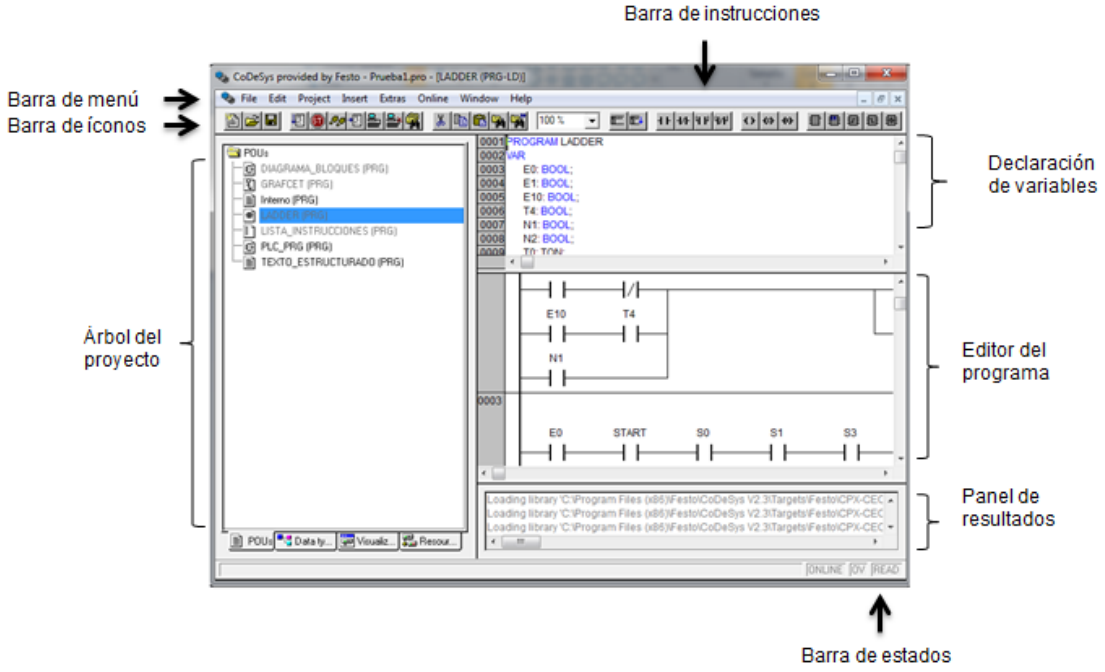

Figura 3 Ventana general Codesys V2.3

Para crear un nuevo POU en el proyecto en la ventana principal se ubica en el árbol del proyecto y se hace clic derecho en la carpeta *POU* tal como se aprecia en la figura 4.

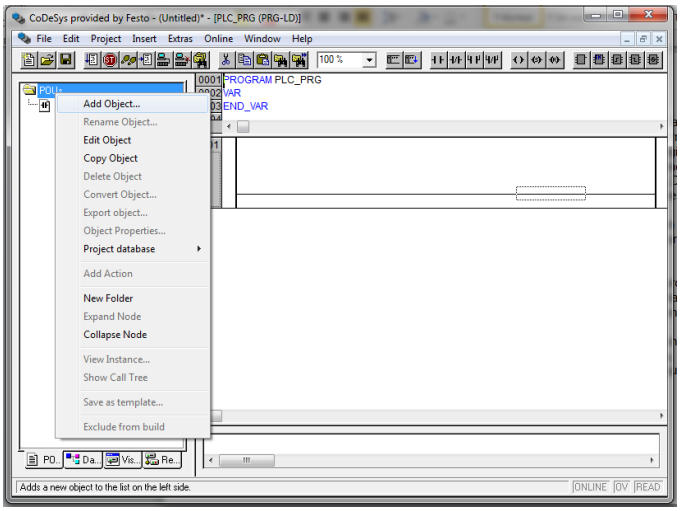

Figura 4 Crear nuevo POU

Para la programación, Codesys cuenta con una librería muy amplia la cual se encuentra en la barra de menú>>*Window*>>*Library Manager*. En la librería se pueden apreciar unas herramientas muy útiles como son contadores, temporizadores, funciones especiales, entre otros.

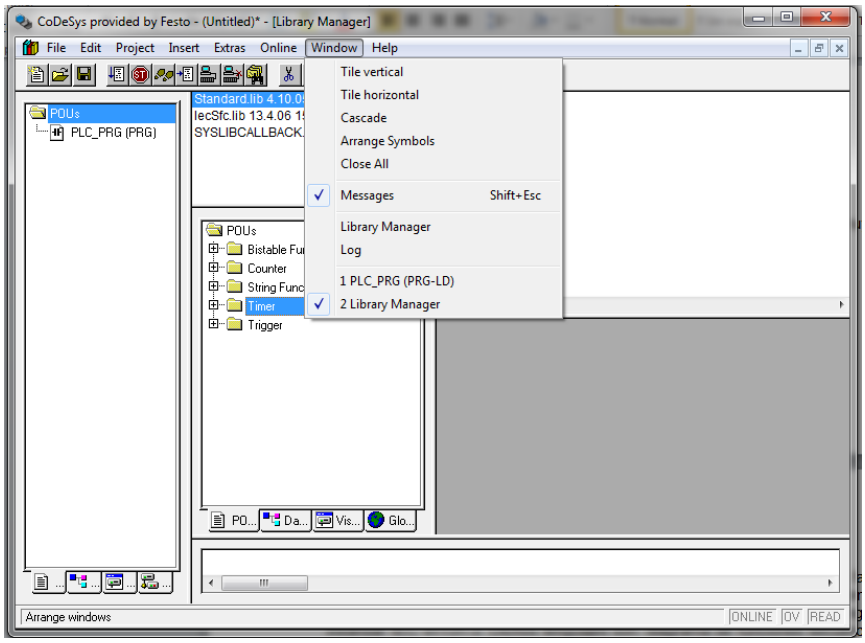

Figura 5 Librería Codesys V2.3

Para la solución del problema de este proyecto de grado se hizo uso de la librería con elementos como son el contador CTU (figura 6), el temporizador TON (figura 7) y el bloque de función BLINK (figura 8).

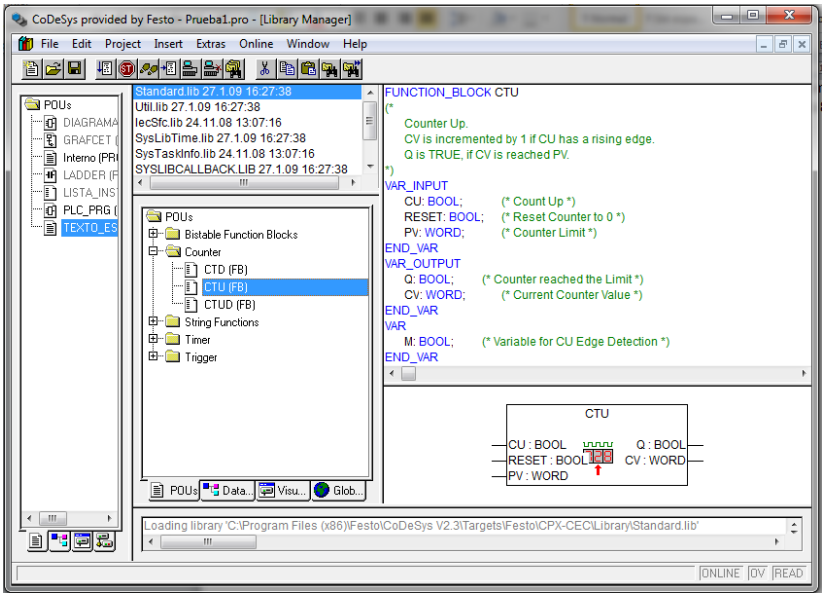

Figura 6 Contador CTU

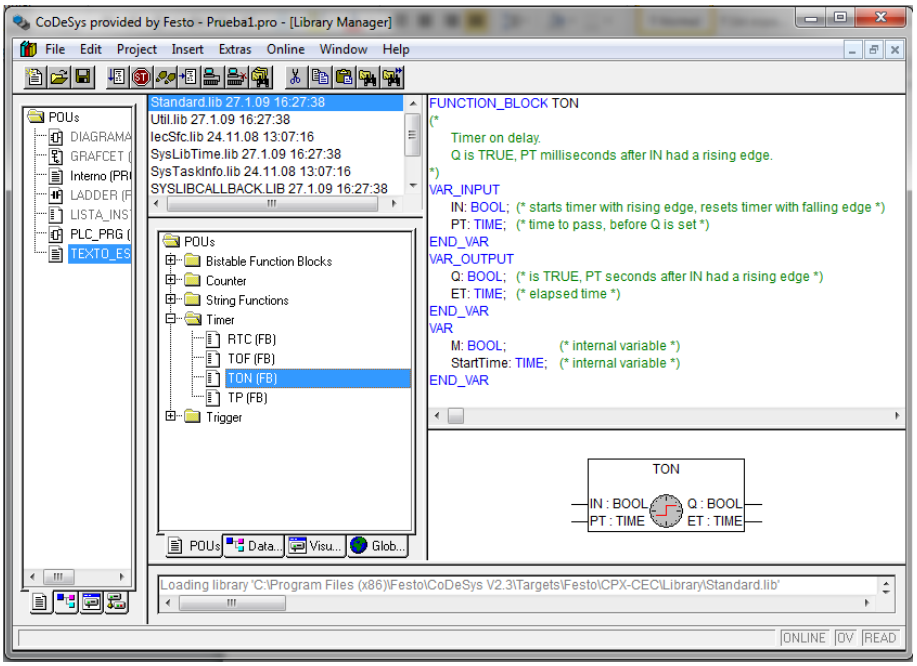

Figura 7 Temporizador TON

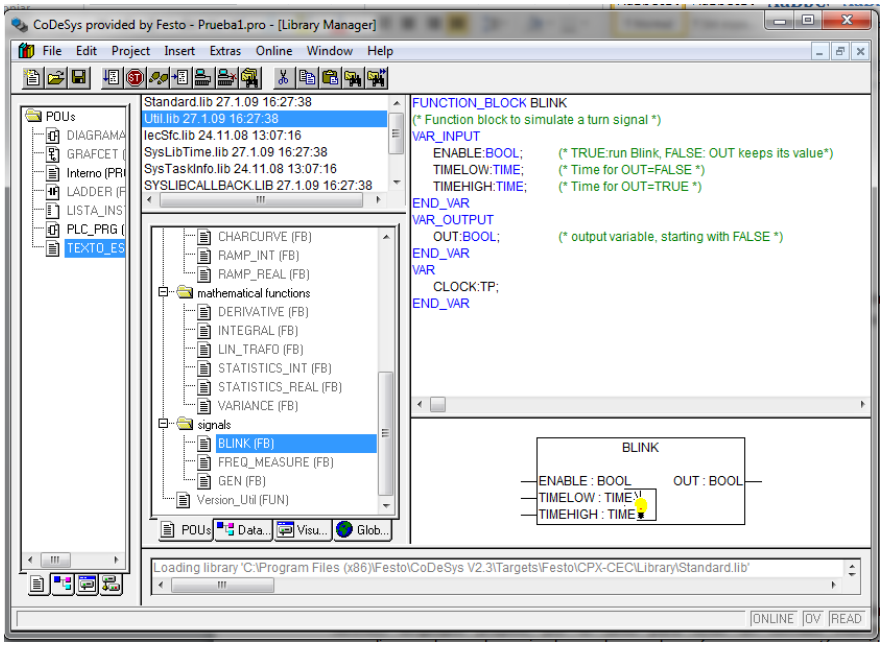

Figura 8 Bloque de función BLINK

Además de la programación de los lenguajes del estándar IEC 61131-3, Codesys V2.3 permite hacer una visualización para observar el proceso en el que se encuentra el proyecto.

Para crear la visualización, se debe dar clic en la pestaña de visualización en la parte inferior izquierda y se agrega un nuevo objeto, como se muestra en la figura 9.

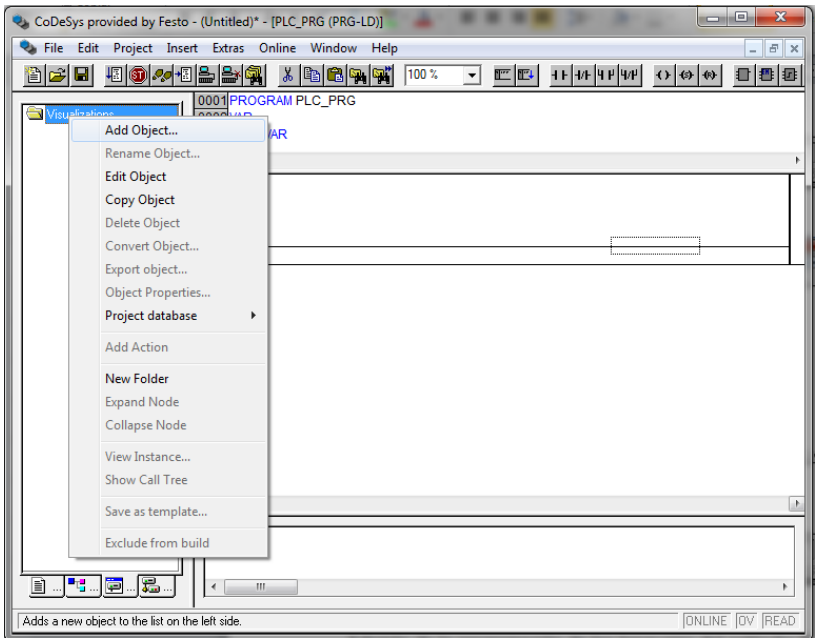

Figura 9 Crear visualización

En la visualización se pueden crear figuras geométricas como rectángulos, elipses, líneas, entre otros y, se puede acceder a herramientas como botones, indicadores, graficadores de tendencias, entre otros. Adicionalmente se pueden cambiar algunos parámetros como son los colores y las variables asociadas a la figura. Para el caso del cambio de colores se debe dar clic en la opción de "*Colors*" y se puede seleccionar el color tanto en estado inactivo en la parte superior llamada "*Color*", y también cuando el botón se encuentra activo en la parte inferior llamada "Alarm color".

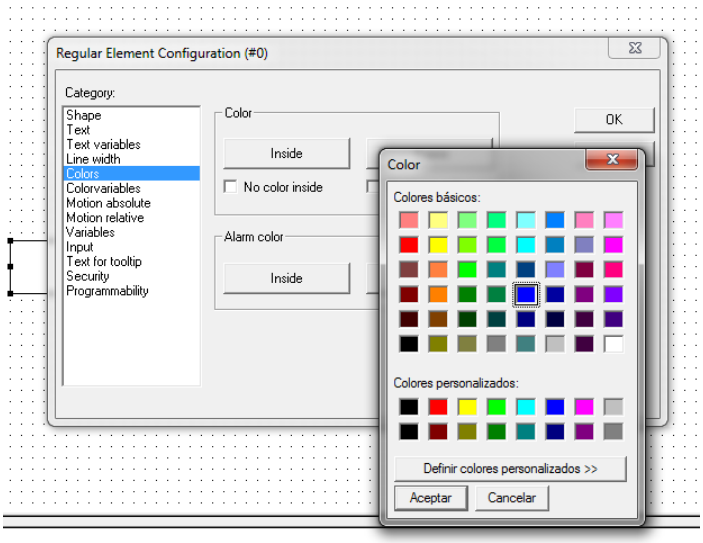

Figura 10 Creación rectángulo y cambio de colores

Para la activación del cambio de color se hace clic en la opción "Variables", y enseguida en "Change color" se escribe el nombre de la señal de activación en cada uno de las salidas (figura 10). De esta manera se crea un dibujo sencillo con sus respectivas variables (figura 11).

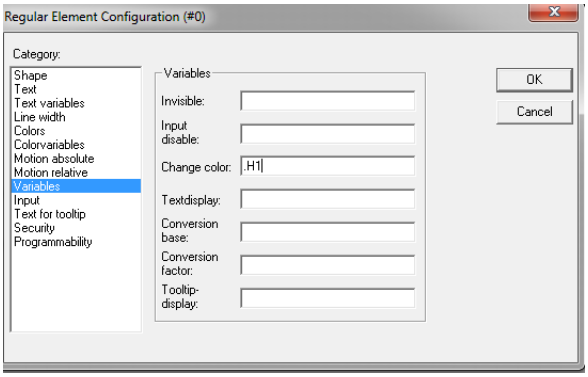

Figura 11 Variable para el cambio de color

A continuación se muestra un ejemplo de programación en Codesys. El problema (figura 12) consiste en tomar las piezas troqueladas de una banda transportadora, rectificar la forma y entregar las piezas rectificadas a través de otra banda transportadora; De manera más precisa, la pieza inicia su recorrido por la banda transportadora1 hasta ser detectado por el sensor SB1, hecho esto, se debe activar un conjunto de actuadores neumáticos que toman la pieza metálica, dicho conjunto posee un actuador giratorio que permite realizar dos movimientos (SP1 y SP2); El primer movimiento que debe hacer el conjunto de actuadores neumáticos es retraer el vástago del

cilindro1 hasta llegar al final de su recorrido y activar el sensor S4, luego la pinza neumática sujeta la pieza, se extiende el vástago del cilindro1 hasta ser detectado por el sensor S3, posteriormente se mueve el actuador giratorio desde la posición SP1 hasta donde se encuentra la troqueladora (posición SP2), allí, se retrae el vástago del cilindro1 nuevamente y la pinza neumática suelta la pieza en la troqueladora, hecho esto, el cilindro1 extiende el vástago y el actuador giratorio regresa a la posición SP1, finalmente se activa el cilindro de la troqueladora, extendiendo su vástago hasta activar el sensor S2 y retrae el vástago nuevamente, como consecuencia cae la pieza metálica en la banda transportadora2 que es activada por el sensor de posición SB2 y finaliza el recorrido al llegar a la posición SB3.

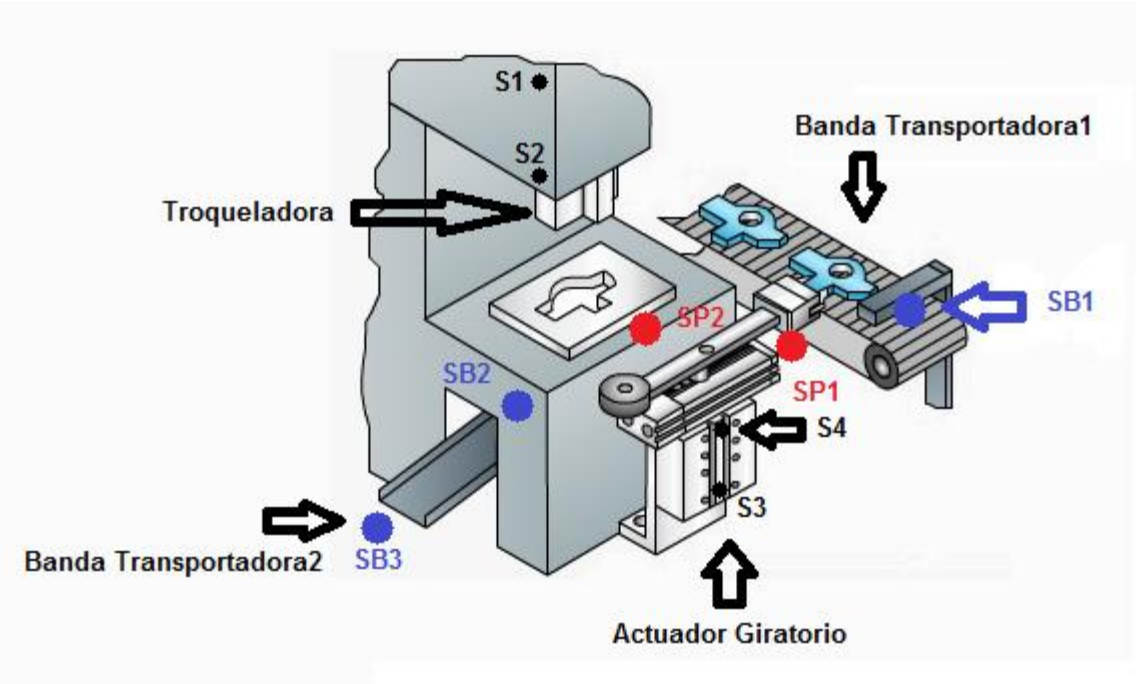

Figura 12 Proceso del problema de ejemplo

A continuación se resuelve el problema mencionado anteriormente usando los lenguajes del estándar IEC 61131-3, además se crea una visualización (Figura 13) para observar las secuencias de operación.

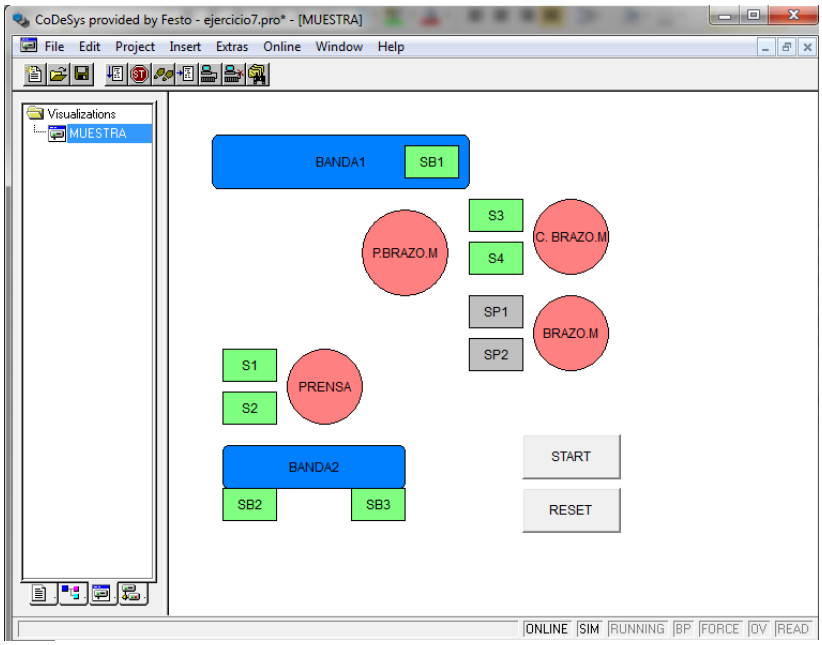

Figura 13 Visualización del ejemplo de simulación en Codesys V2.3

Para el desarrollo del automatismo, se inicia haciendo el modelado en GRAFCET y se obtienen las ecuaciones, la selección de esta herramienta de modelado se deba a que el conjunto de ecuaciones obtenidas permiten desarrollar la lógica en cualquier lenguaje de programación. Aprovechando que el lenguaje de programación SFC tiene relación con esta herramienta de modelado se ha desarrollado la programación en dicho lenguaje (figura 14) y se elaboran las ecuaciones para pasar con mayor facilidad a otros lenguajes de programación.

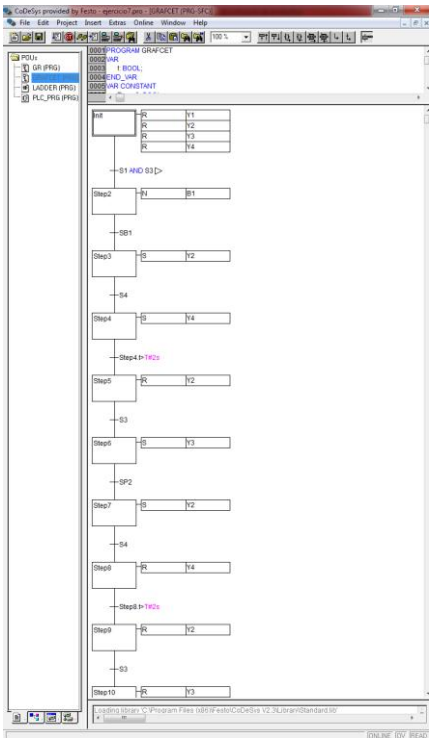

Figura 14 Lenguaje de programación SFC o GRAFCET

La siguiente es la fórmula general que permite pasar del modelo de GRAFCET a cada uno de los lenguajes del estándar IEC 61131-3 de una manera sencilla.

Etapa Activa=Etapa anterior * Transición anterior + Etapa actual * Etapa siguiente negada (Ecuación 1).

Recordando que las operaciones de suma (+) y multiplicación (*) son operaciones lógicas y no aritméticas; Por tanto, en las figuras 15, 16, 17 y 18 podemos observar fragmentos del desarrollo del modelo en otros lenguajes de programación contenidos en la norma.

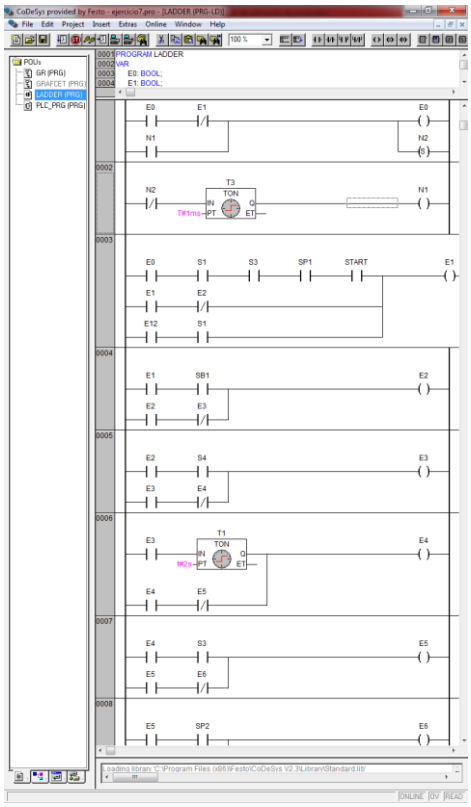

Figura 15 Lenguaje de programación LD o Ladder

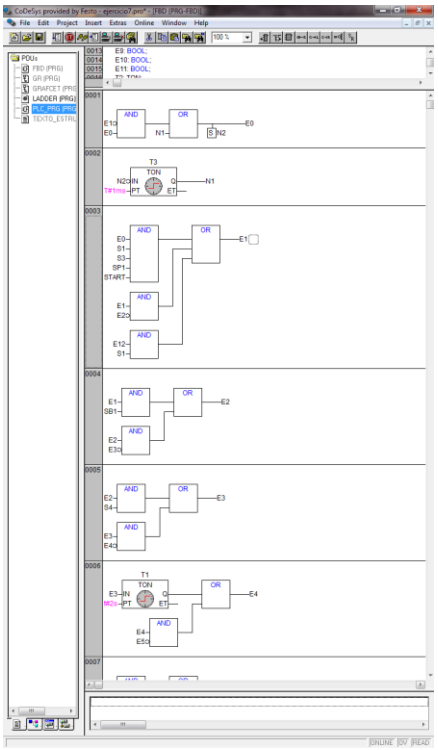

Figura 16 Lenguaje de programación FBD

Figura 17 Lenguaje de programación ST

Figura 18 Lenguaje de programación IL

Por otro lado, Fluidsim es un software de la empresa Festo que permite realizar el diseño y simulación de circuitos neumáticos, hidráulicos y eléctricos siendo una herramienta práctica para la formación de profesionales y las empresas de ingeniería. Los parámetros de los componentes del Fluidsim son iguales a los de Festo Didactic, además que sus componentes se pueden adaptar fácilmente a las características de otros componentes.

La creación de un nuevo proyecto es muy sencillo en la versión Fluidsim P, basta con hacer clic en “Archivo” y luego en "nuevo” o presionando las teclas “Ctrl+N”; La librería de símbolos se localiza en la parte izquierda de la ventana principal (figura 19), en ella se encuentra el contenido completo y compartido para los circuitos neumáticos, hidráulicos y eléctricos.

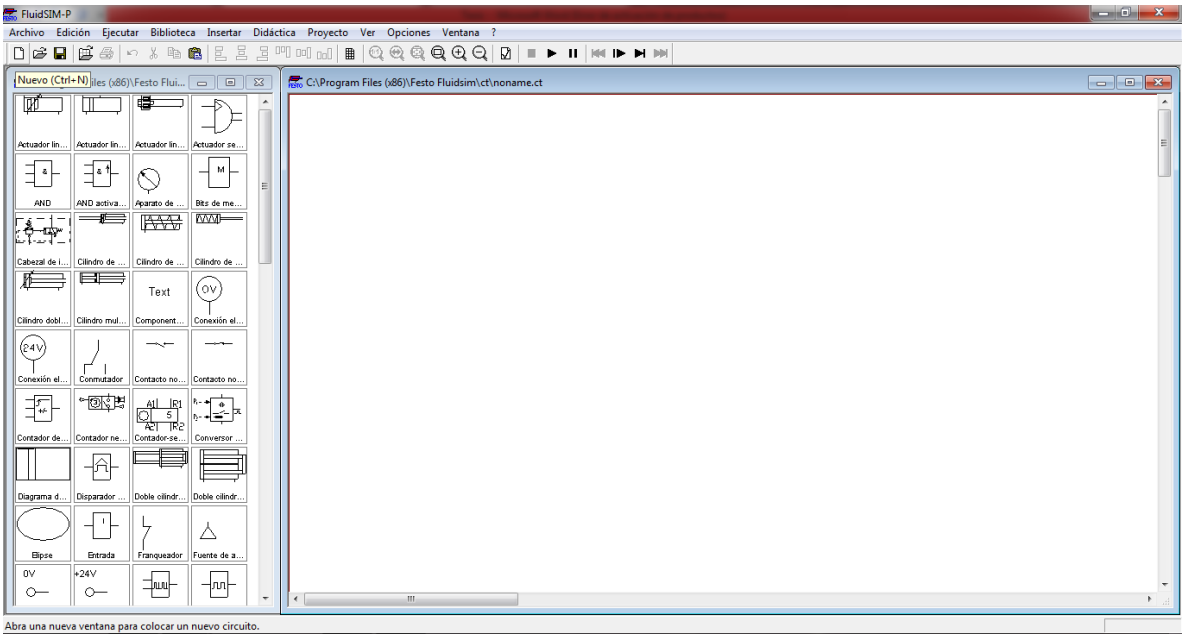

Figura 19 Fluidsim

3. Objetivos

3.1 Objetivo General

- Realizar un análisis comparativo de los diferentes lenguajes de programación de PLC normalizados en la norma IEC61131-3.

3.2 Objetivos específicos

- Estudiar los lenguajes de programación normalizados según el estándar IEC61131-3.
- Resolver un problema de automatización secuencial y combinacional utilizando cada uno de los lenguajes.
- Implementar la solución del problema secuencial y combinacional utilizando el software Codesys.
- Programar un PLC para resolver un problema secuencial y combinacional en un sistema electro-neumático.
- Realizar un análisis comparativo de los lenguajes de programación de PLC con base en la norma IEC 61131-3.

4. Planteamiento del problema

En la actualidad no se ha hecho ningún estudio comparativo de los lenguajes de programación de PLC contenidos en la norma IEC 61131-3 y sobre cuáles son los beneficios reales que trae programar en un lenguaje u otro. Por este motivo, en el presente proyecto de grado se busca hacer un análisis desde diferentes enfoques, incluyendo el tiempo de descarga del programa al PLC y el tiempo del ciclo de SCAN del PLC ya que si estos tiempos varían significativamente de un lenguaje a otro y considerando que la selección del lenguaje puede representar cambios en la productividad, en los tiempos de desarrollo y de mantenimiento.

Para el logro del objetivo general es necesario considerar las ventajas, desventajas y características de cada lenguaje de programación y con ello, observar el ciclo de SCAN, el cual es un procedimiento que realiza el PLC de manera repetitiva una vez entra en funcionamiento, dicho procedimiento se basa en realizar de manera cíclica las operaciones de lectura del estado de las entradas tanto análogas como digitales, lectura de las líneas del programa y actualización del estado de las salidas. El tiempo que demora en ejecutar todas estas operaciones se le conoce como el tiempo del ciclo de SCAN. Por otro lado, se observará el tiempo que tarda en transferir el firmware del computador al PLC.

Para este proyecto de grado y en función del objetivo general se diseñó un ejercicio que considerara la mayor cantidad de características posibles en problemas reales de automatización como estructuras secuenciales, combinacionales, convergentes y divergentes con el ánimo de obtener resultados concluyentes y alcanzar los objetivos de manera satisfactoria.

Con base en lo anterior, se expone el siguiente ejercicio: Para comenzar el programa se debe de verificar que los vástagos de los cilindros estén retraídos y que los pilotos se encuentren apagados, una vez confirmado se presiona el botón START, y al activarse el sensor S1 (sensor de posición que comprueba que hay una pieza) se da inicio al proceso. Hecho esto, avanza el vástago del cilindro 1.0 y queda extendido hasta que temporice el cuarto temporizador e inmediatamente se enciende el piloto H1 intermitentemente. Una vez se confirme que avanzó el cilindro 1.0, se desarrollan 2 actividades en simultáneo. La primer actividad comienza con el

avance el vástago del cilindro 2,0, y una vez haya avanzado este comienza a retroceder, asimismo avanza el vástago del cilindro 3.0 y temporiza 8 segundos. Cuando se confirme que avanzó el cilindro 3.0, que temporizó los 8 segundos y se retrae el vástago del cilindro 2.0, se pone en marcha el retroceso vástago del cilindro 3.0 y hace un conteo. Cuando el contador confirme que se ha realizado este proceso 5 veces se enciende el piloto H2 y se apaga H1, por el contrario si no se ha hecho el proceso 5 veces, vuelve y arranca la primera actividad. Por otro lado, la segunda actividad inicia con el avance del vástago del cilindro 4.0, una vez se confirme que avanzo el vástago retrocede. Si el contador del primer proceso tiene un conteo menor o igual a 3 se hace una temporización de 3 segundos y vuelve a iniciar el segundo proceso, pero si el contador es mayor a 3 se enciende piloto H3 y hace una temporización de 3 segundos el cual confirma que el segundo proceso ha finalizado completamente. Por último, cuando hayan finalizado los 2 procesos, es decir que los pilotos H2 y H3 se encuentren encendidos, H1 apagado y se haya temporizado los 3 segundo del segundo proceso el vástago del cilindro 1.0 retrocede e inmediatamente se activa un cuarto temporizador el cual tiene un tiempo de 5 segundos

5. Solución del automatismo

La solución del automatismo se divide en dos partes: La primera parte consiste en hacer una simulación del desarrollo de la solución en cada uno de los lenguajes del estándar IEC 61131-3 haciendo uso del software Codesys V2.3. La segunda parte se basa en realizar pruebas sobre un PLC real, en el laboratorio del SENA – Centro de Diseño e Innovación Tecnológica Industrial de Dosquebradas, implementando cada uno de los lenguajes de programación en un PLC Festo CPX-CEC-C1; Para el uso de actuadores y sensores, se usó el software Fluidsim y un OPC (OLE for Process Control) integrado en el software Fluidsim, es así como se aprovecharán las bondades de este software para realizar la simulación del circuito electro-neumático y por ende, de las acciones que se deben ejecutar como respuesta al ejercicio planteado.

De acuerdo con la primera parte de la solución, se realizó un esbozo del planteamiento del ejercicio a través de la visualización que brinda Codesys V2.3, creando entonces los 3 pilotos H1, H2, H3, el botón Start/Stop el cual sirve para dar arranque al programa, un botón S0 que funciona como un sensor de posición para saber si hay una pieza y poder comenzar con la operación y 4

cilindros que se diseñaron con ayuda de imágenes en internet y programas como Fluidsim, dando como resultado una plantilla como la que se muestra en la figura 20.

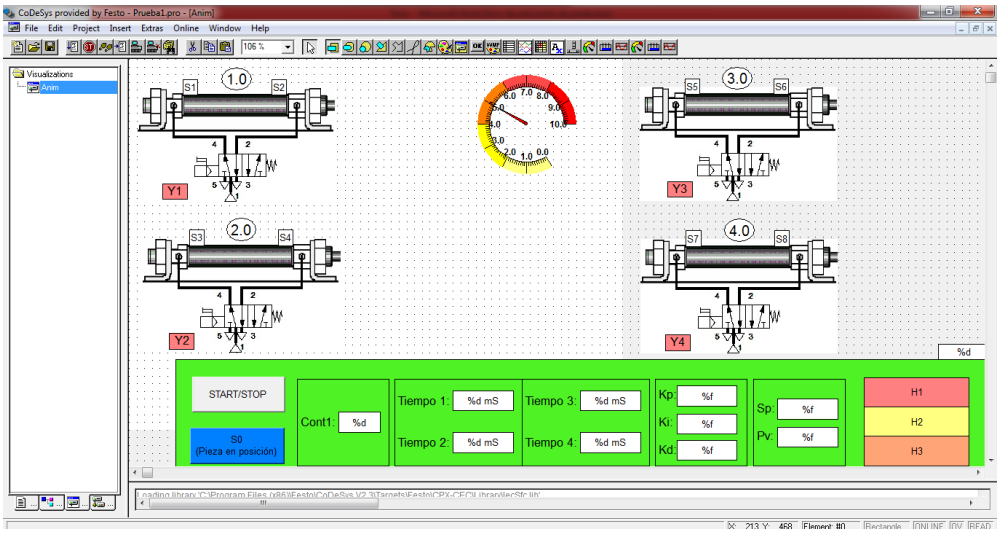

Figura 20 Visualización de la simulación

Se desarrolló toda programación para la animación de la simulación y se procedió al modelado en GRAFCET (figura 21), hecho esto, se elaboran las ecuaciones (haciendo uso de la ecuación 1) que permiten la generación de la solución en cualquier lenguaje de la norma IEC 61131-3 de forma fácil y rápida.

En Codesys V2.3 el lenguaje de programación SFC tiene una opción muy interesante que admite tener las acciones al lado de la etapa o dentro de la etapa. Para este proyecto se realizó la programación dentro de la etapa abriendo la posibilidad de realizar la programación en otros lenguajes y de esta manera hacerlo mucho más sencillo. En otras palabras en el software Codesys V2.3 por medio del lenguaje SFC se puede hacer una mezcla de lenguajes de programación.

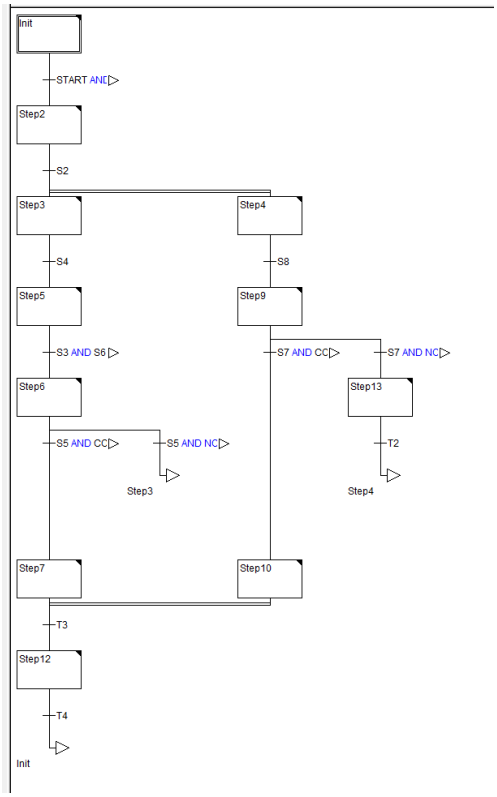

Figura 21 Programación en GRAFCET

Por facilidad en la programación de las acciones se seleccionó el lenguaje LD como mejor alternativa, en la figura 22 se muestra la acción incluida a la etapa 5 y el uso del lenguaje tipo escalera.

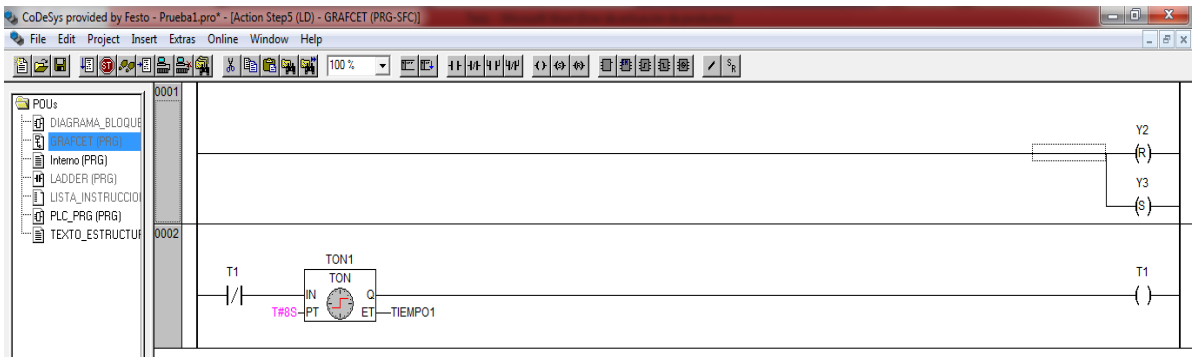

Figura 22 Acciones de la etapa 5

Haciendo uso de las ecuaciones de todas las etapas se hace la respectiva programación en cada uno de los diferentes lenguajes de programación del estándar IEC 61131-3, considerando las

especificaciones de cada lenguaje y del software Codesys V2.3, las cuales pueden cambiar de acuerdo a la versión o al software, y se comprueba que todos operen correctamente.

A continuación se muestra en las figuras 23, 24, 25, y 26 fragmentos de la programación realizada con cada uno de los lenguajes del estándar IEC 61131-3, todos desarrollados en el software Codesys V2.3

```
0020 IF (E1=TRUE AND S2=TRUE OR E2=TRUE AND E3=FALSE OR E4=TRUE AND S5=TRUE AND CONT1=FALSE)
0021 THEN E2:=TRUE;
0022 ELSE E2:=FALSE;
0023 END_IF
0024 (*E3=E2*S4+E3*E4' *)
0025 IF (E2=TRUE AND S4=TRUE OR E3=TRUE AND E4=FALSE)
0026 THEN E3:=TRUE;
0027 ELSE E3:=FALSE;
0028 END_IF
0029 (*E4=E3*S3*S6*T1+E4*E5'*E2' *)
0030 IF (E3=TRUE AND S3=TRUE AND S6=TRUE AND T1=TRUE OR E4=TRUE AND E5=FALSE AND E2=FALSE)
0031 THEN E4:=TRUE;
0032 ELSE E4:=FALSE;
0033 END_IF
0034 (*E5=E4*S5*CONT1+E5*E10' *)
0035 IF (E4=TRUE AND S5=TRUE AND CONT1=TRUE OR E5=TRUE AND E10=FALSE)
0036 THEN E5:=TRUE;
0037 ELSE E5:=FALSE;
0038 END_IF
0039 (*E6=E1*S2+E6*E7'+E8*T2 *)
0040 IF (E1=TRUE AND S2=TRUE OR E6=TRUE AND E7=FALSE OR E8=TRUE AND T2=TRUE)
0041 THEN E6:=TRUE;
0042 ELSE E6:=FALSE;
0043 END_IF
0044 (*E7=E6*S8+E7*E9'*E8' *)
0045 IF (E6=TRUE AND S8=TRUE OR E7=TRUE AND E9=FALSE AND E8=FALSE)
0046 THEN E7:=TRUE;
0047 ELSE E7:=FALSE;
0048 END_IF
0049 (*E8=E7*S7*CONT2'+E8*E6' *)
0050 IF (E7=TRUE AND S7=TRUE AND CONT2=FALSE OR E8=TRUE AND E6=FALSE)
0051 THEN E8:=TRUE;
0052 ELSE E8:=FALSE;
0053 END_IF
0054 (*E9=E7*S7*CONT2+E9*E10' *)
0055 IF (E7=TRUE AND S7=TRUE AND CONT2=TRUE OR E9=TRUE AND E10=FALSE)
0056 THEN E9:=TRUE;
0057 ELSE E9:=FALSE;
0058 END_IF
0059 (*E10=E5*E9*T3+E10*E0' *)
0060 IF (E5=TRUE AND E9=TRUE AND T3=TRUE OR E10=TRUE AND E0=FALSE)
0061 THEN E10:=TRUE;
0062 ELSE E10:=FALSE;
0063 END_IF
0064 (*RESET  Y4=RESET Y3=RESET Y2=RESET Y1=RESET H3=RESET H2=RESET H1=E0*)
0065 IF E0=TRUE
0066 THEN Y4:=FALSE; Y3:=FALSE; Y2:=FALSE; Y1:=FALSE; H3:=FALSE; H2:=FALSE; H1:=FALSE;
0067 END_IF
0068 (*SET Y1=E1*)
0069 IF E1=TRUE
0070 THEN Y1:=TRUE;
0071 END_IF
0072 (*SET Y2=E2*)
0073 IF E2=TRUE
0074 THEN Y2:=TRUE;
0075 END_IF
0076 (*SET Y3=RESET Y2=E3*)
0077 IF E3=TRUE
0078 THEN Y3:=TRUE; Y2:=FALSE;
0079 END_IF
0080 (*T1=E3*TON1 *)
0081 IF E3=TRUE
0082 THEN K1:=TRUE;
0083 ELSE K1:=FALSE;
0084 END_IF
0085 TON1(IN:=K1, PT:=T#8S);
0086 T1:=TON1.Q;
0087 TIEMPO1:=TON1.ET;
```

Figura 23 Programación en texto estructurado

En la codificación del lenguaje de programación lista de instrucciones se debe tener en cuenta una de las condiciones propias del lenguaje en Codesys V2.3 cuando se tiene una suma booleana con más de 3 variables es que a partir de la tercera variable se debe colocar un paréntesis luego de la función OR, hasta donde culmina la suma lógica. En la figura 24 se muestra una parte del

código donde se aprecia una suma booleana con 3 variables en la ecuación E0 y E2 respectivamente.

```
(*N1=N2'*T0*)
LDN     N2
ST      N3
CAL     T0 (IN:=N3, PT:= T#1MS)
LD      T0.Q
ST      N1
(*E0=SET N2=E0*E1'+N1+E10*T4 *)
LD      E0
ANDN    E1
OR      N1
OR      (E10
AND     T4
)
ST      E0
S       N2
(*E1=E0*START*S1*S3*S5*S7+E1*E2'*E6' *)
LD      E0
AND     START
AND     S0
AND     S1
AND     S3
AND     S5
AND     S7
OR      E1
ANDN    E2
ANDN    E6
ST      E1
(*E2=E1*S2+E2*E3'+E4*S5*CONT1' *)
LD      E1
AND     S2
OR      E2
ANDN    E3
OR      (E4
AND     S5
ANDN    CONT1
)
ST      E2
(*E3=E2*S4+E3*E4' *)
LD      E2
AND     S4
OR      E3
ANDN    E4
ST      E3
(*E4=E3*S3*S6*T1+E4*E5'*E2' *)
LD      E3
AND     S3
AND     S6
AND     T1
OR      E4
ANDN    E5
ANDN    E2
ST      E4
(*E5=E4*S5*CONT1+E5*E10' *)
LD      E4
AND     S5
AND     CONT1
OR      E5
ANDN    E10
ST      E5
(*E6=E1*S2+E6*E7'+E8*T2 *)
LD      E1
AND     S2
OR      E6
ANDN    E7
OR      (E8
AND     T2
)
```

Figura 24 Programación en lista de instrucciones

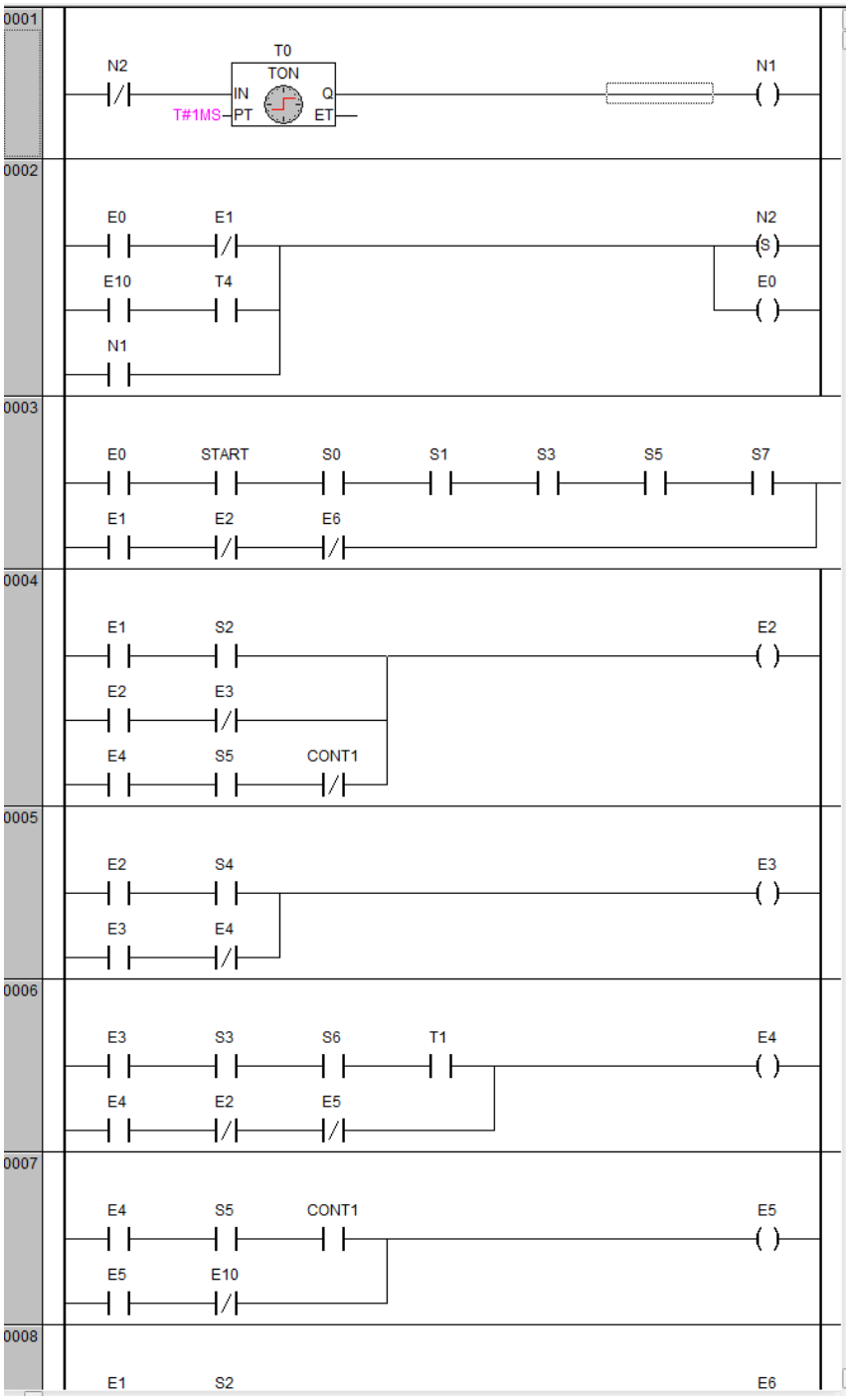

Figura 25 Programación en Ladder

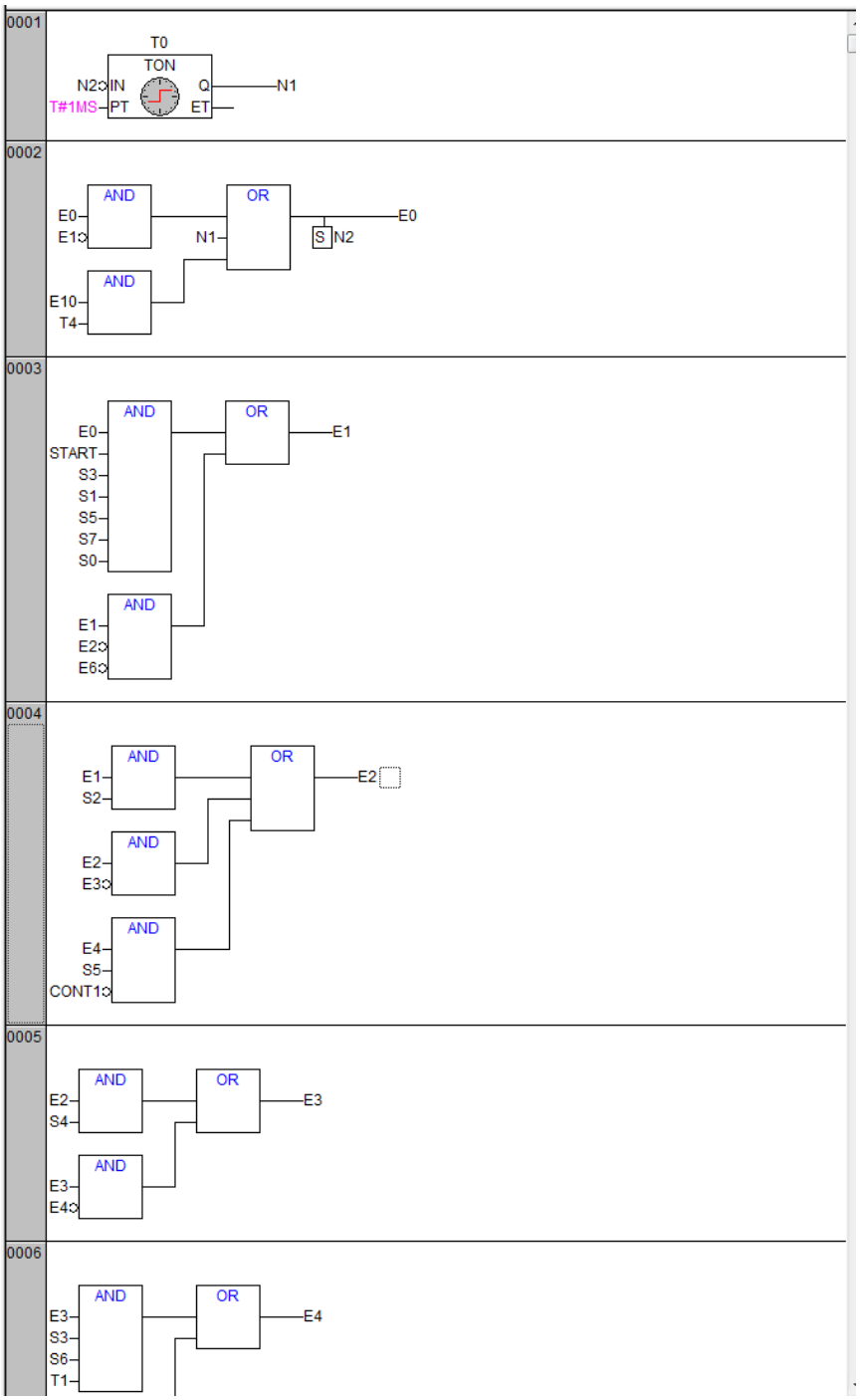

Figura 26 Fragmento del código en diagrama de bloques

Para hacer un cambio de lenguajes de forma rápida y permitir que la visualización opere de manera correcta los programas deben tener un POU de control el cual es el programa principal (PLC_PRG), y cada uno de los lenguajes son programas auxiliares. En el programa principal se

realiza la programación del bloque de función BLINK y de los contadores, de igual manera se tiene el programa interno y la selección de cada lenguaje, como se aprecia en la figura 27.

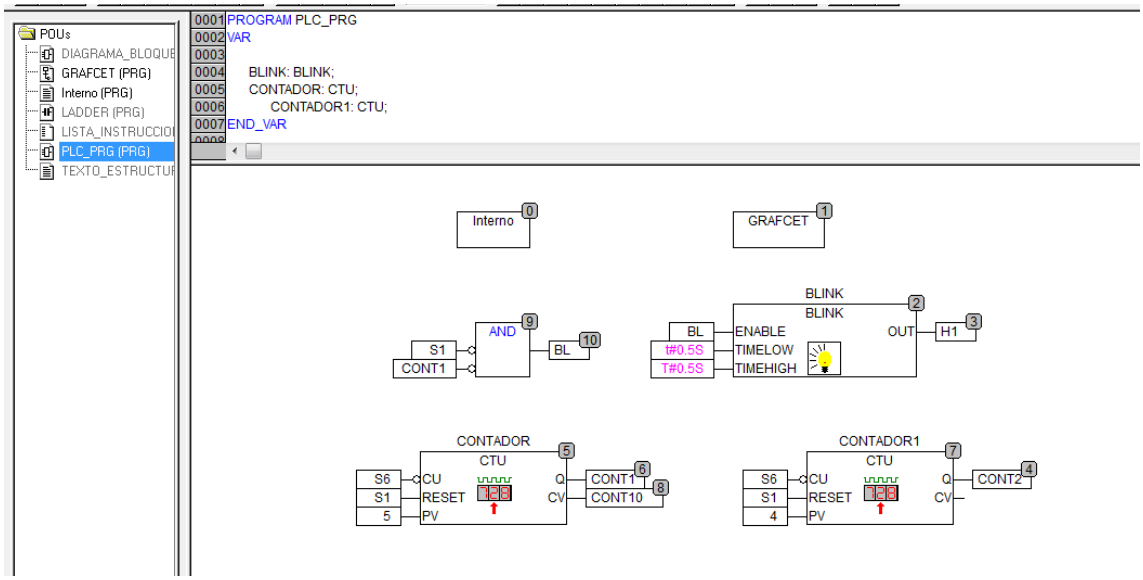

Figura 27 Programa principal PLC_PRG

Una vez comprobado que las simulaciones funcionaron correctamente, se procedió a realizar las pruebas con un PLC Festo CPX-CEC-C1 (figura 28) en el laboratorio del SENA – Centro de Diseño e Innovación Tecnológica Industrial de Dosquebradas.

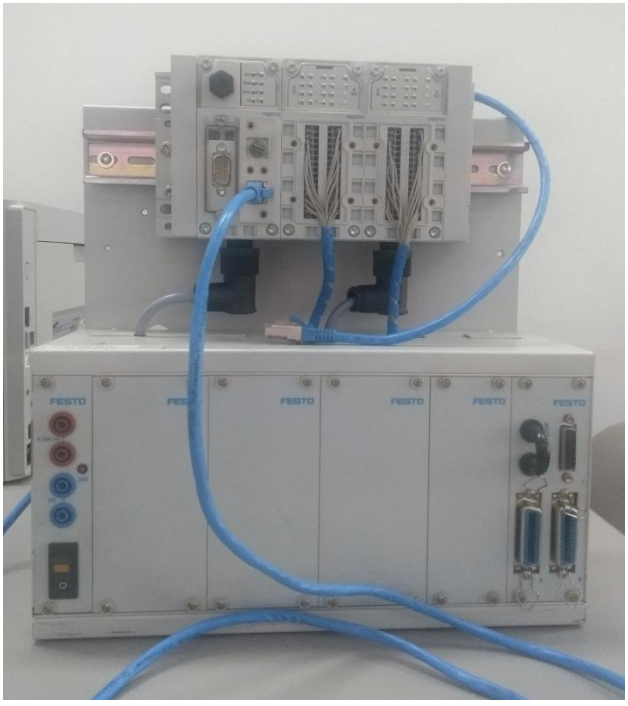

Figura 28 PLC Festo CPX-CEC-C1

Para que el PLC reconozca las señales tanto de entrada como de salida se hizo un ajuste al programa principal, borrando el bloque de llamado del programa "interno" y a cambio se ponen bloques de entrada y salida, dando como resultado un programa como el que aparece en la figura 29.

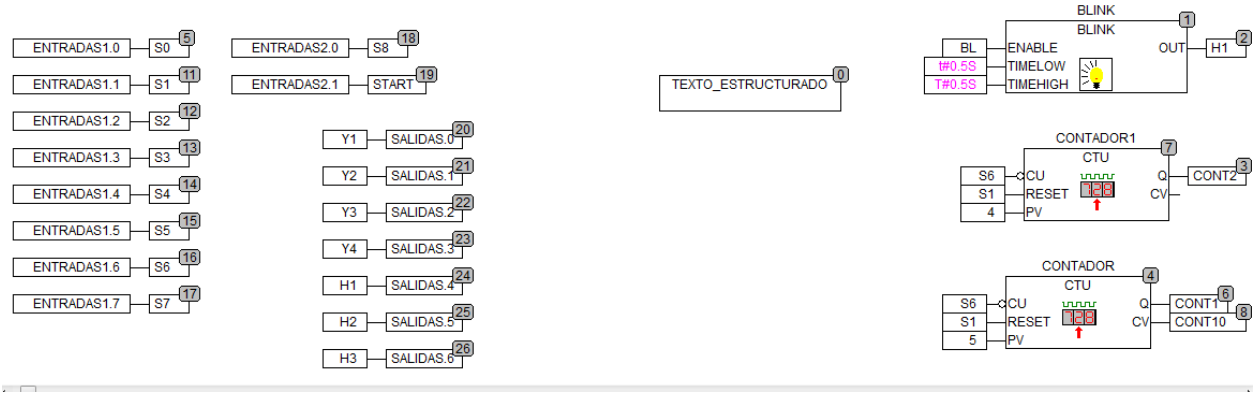

Figura 29 Programación en el programa PLC_PRG

Las pruebas se realizaron conectando el PLC con el simulador Fluidsim a través del OPC, ya que el software describirá un comportamiento similar al de los cilindros reales, considerando que la velocidad de activación la brinda el PLC y no el simulador. En el software Fluidsim se diseñaron los circuitos neumáticos y eléctricos atribuyendo parámetros a los componentes similares a los de algunos elementos reales para que los resultados obtenidos con el Fluidsim sean similares a los resultados con componentes reales.

Para el circuito neumático se usaron los siguientes elementos: fuente de aire comprimido, unidad de mantenimiento, válvulas de 5/2 vías (para esta válvula se le hace un modificación en el accionamiento izquierdo haciendo uso de la opción pilotado y cambiando el esfuerzo y la parte neumático eléctrica), cilindros de doble efecto diferencial con amortiguación regulable, y válvulas reguladoras de caudal unidireccional con un grado de apertura del 70%. Para el circuito eléctrico se necesitaron elementos como una fuente de tensión de 24V y 0V, interruptor de alimentación óptica (Sensor óptico), puerto de salida Fluidsim, obturador (final de carrera – interruptor mecánico), puerto de entrada Fluidsim, solenoides de electroválvulas e indicadores luminosos; Todos estos componentes y sus conexiones se pueden observar en la figura 30.

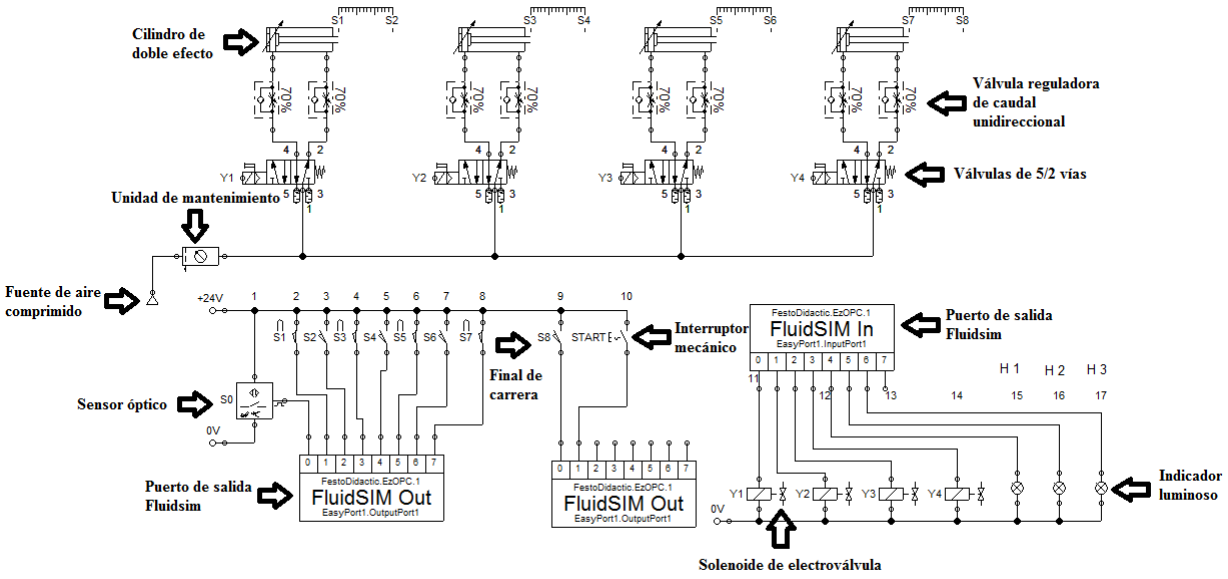

Figura 30 Visualización circuito neumático y eléctrico en Fluidsim

6. Análisis de resultados

Una vez finalizada la implementación en el PLC CPX-CEC-C1, se determinaron los tiempos de descarga del firmware al PLC y del ciclo de SCAN con el fin de analizar cambios sustanciales en dichos tiempos que permitan inferir en la selección de un lenguaje de programación de manera objetiva.

Las herramientas y módulos que ofrece el software Codesys V2.3 no permiten obtener los tiempos del ciclo de SCAN del PLC por este motivo no es posible por el momento medir dicho tiempo para la ejecución de los programas desarrollados en cada lenguaje, tampoco es posible medir la cantidad de instrucciones generadas antes de ser enviadas al PLC, considerando que no todos los lenguajes son textuales y que de obtener este valor se podría estimar el tiempo basados en el manual del fabricante que menciona que el tiempo de ejecución se puede determinar cómo 200µs por cada 1k de instrucciones.

Por otro lado, se hicieron varias pruebas del tiempo de descarga de los programas al PLC. Los datos obtenidos del tiempo de descarga se observan en la tabla 1.

Lenguaje	Prueba 1	Prueba 2	Prueba 3	Prueba 4	Promedio
Texto Estructurado	2,7	2,24	2,39	2,45	2,445
Lista de Instrucciones	2,92	2,34	2,13	2,31	2,425
Ladder	2,62	2,27	2,6	2,57	2,515
GRAFCET	2,42	2,42	2,47	2,65	2,49
Diagrama de Bloques	2,8	2,4	2,3	2,28	2,445
				Total	2,464

Tabla1. Resultados tiempo de descarga de los programas al PLC en segundos.

Durante la implementación de la solución en cada lenguaje se pudo obtener información de interés que puede ser algo subjetiva teniendo en cuenta que se realizó con un software en particular y que el trabajo fue desarrollado por una persona. De este ejercicio se obtuvieron los siguientes resultados (Tabla 2):

	Bloque de función Temporizador	Bloque de función Contador	Bloque de función Especial	Problema Secuencial	Problema Combinacional
SFC	Intermedio	Complejo	Complejo	Sencillo	Sencillo
LD	Sencillo	Sencillo	Sencillo	Sencillo	Sencillo
FBD	Sencillo	Sencillo	Sencillo	Sencillo	Sencillo
ST	Sencillo	Sencillo	Complejo	Sencillo	Sencillo
IL	Sencillo	Sencillo	Complejo	Sencillo	Intermedio

Tabla2. Comparación Lenguajes del estándar IEC 61131-3 Vs Características.

7. Conclusiones

- La programación en lenguajes como LD y FBD es muy sencilla dado que son lenguajes gráficos y una de sus principales ventajas es que permite reconocer fácilmente la ubicación de las señales de entrada y salida en los bloques de funciones como son los temporizadores, contadores y bloques de funciones especiales.

- Los lenguajes de programación textual pueden ser más atractivos para las personas que dominan un lenguajes de programación para computadores, microprocesadores o microcontroladores; Particularmente para una persona que este acostumbrada a programar en lenguajes como C++, Matlab, entre otros, será muy sencillo realizar la programación en el lenguaje de texto estructurado y, para las personas que programan con lenguajes de bajo nivel puede ser muy interesante programar en IL.

- Como ejemplo, el bloque de función BLINK el cual es un componente del software Codesys cuya salida funciona como una señal de reloj con tiempos de estados altos y bajos configurables de manera independiente, este es muy útil cuando se necesita alguna señal intermitente (piloto H1 del problema resuelto en este proyecto). La programación del bloque de función BLINK en lenguajes como texto estructurado, lista de instrucciones resulta complejo de programar debido a que la sintaxis no es clara, ni regular para cada bloque de función.

- En el lenguaje de programación SFC cabe considerar que si se usan bloques de función especial será necesario recurrir a otros lenguajes de programación para realizar las acciones necesarias; En este proyecto de grado se tuvo inconvenientes a la hora de restaurar el contador, dado que el RESET no podía activarse en cualquier etapa por las propiedades del GRAFCET, sin embargo aprovechando la facilidad de programación de estos bloques de función en FBD y LD, se optó por implementar el contador en FBD, además, dado que el programa principal controla los subprogramas, dicha implementación se realizó allí.

- El modelado en GRAFCET para resolver problemas de tipo secuencial y combinacional sigue siendo una alternativa muy interesante por su simplicidad en comparación de otras herramientas de modelado para el desarrollo de automatismos como las Redes de Petri. Adicionalmente, la herramienta de modelado se programó primero en el lenguaje SFC y por medio de la elaboración de las ecuaciones se pasó con mayor facilidad a otros lenguajes de programación.

- La mayor ventaja que tiene el lenguaje de programación diagrama de funciones secuenciales es diseñar soluciones de problemas de tipo secuencial y/o combinacional por las características gráficas de este lenguaje permitiendo visualizar el procedimiento paso a paso y así, hacerle seguimiento a cualquier inconveniente que se tenga, de un modo más simple que en los lenguajes gráficos LD y FBD.

- En el caso del tiempo del ciclo de SCAN del PLC, en este caso en particular no fue posible medirlo con las herramientas que ofrece el software Codesys V2.3, dado que no tiene bloques de función especial para esto, ni banderas, marcas o variables especiales relacionadas con este parámetro.

- Considerando que los tiempos de descarga del firmware al PLC fueron prácticamente constantes independiente del lenguaje de programación, se puede decir que si se tienen problemas de cualquier tipo (secuencial, combinacional, convergente, divergente, entre otros) el lenguaje de programación no determina el tiempo de descarga.

8. Trabajos futuros

Se plantea realizar este ejercicio en autómatas de otras marcas como Siemens, Mitsubishi, entre otros, midiendo el ciclo de SCAN y el tiempo de descarga de los programas al PLC y considerando nuevamente las características, ventajas y desventajas de cada software.

Se recomienda hacer el mismo ejercicio desarrollado en este proyecto de investigación, incluyendo variables análogas, contadores rápidos, comunicaciones y controladores PID para incrementar el grado objetividad de la solución.

9. Bibliografía

[1]. http://herramientas.camaramedellin.com.co/Inicio/Buenaspracticasempresariales/BibliotecaProducci%C3%B3nyOperaciones/Automatizaciondelosprocesosindustriales.aspx

[2]. https://www.festo-didactic.com/es-es/productos/software-e-learning/fluidsim/fluidsim-5.htm?fbid=ZXMuZXMuNTQ3LjE0LjE4LjU5MS43OTc1

[3]. http://www.infoplc.net/blogs-automatizacion/item/101930-ciclo-scan-automata-plc

[4]. https://sites.google.com/a/misena.edu.co/risautomatizacion/plc